PATENT ABSTRACTS OF JAPAN

(11)Publication number : 11-088891
(43)Date of publication of application : 30.03.1999

(51)Int.Cl. H04N 7/32

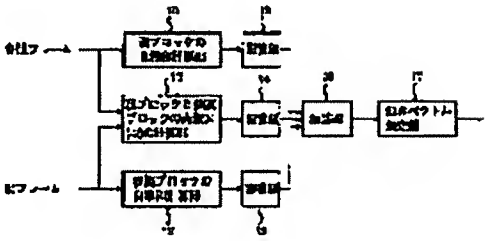
(21)Application number : 10-194893
(22)Date of filing : 10.07.1998

(71)Applicant : NEC CORP
(72)Inventor : NAITO YUKIHIRO
MIYAZAKI TAKASHI
KURODA ICHIRO

(30)Priority
Priority number : 07195998 Priority date : 07.07.1995 Priority country : JP

(54) MOVEMENT DETECTION SYSTEM

(57)Abstract:
PROBLEM TO BE SOLVED: To sharply reduce operation quantity in a block matching type movement detection system for detecting the moving vector of a moving image by using a processor provided with an arithmetic logical computing unit and a product-sum computing unit.



SOLUTION: The square sum of pixel values of a current block which is calculated by a calculation part 10 is stored in a storage part 13. A calculation part 11 calculates a value obtained by multiplying an inner product between a current block and a part or all of reference blocks in a reference area by (-2) in each reference block and stores the calculated value in a storage part 14. A calculation part 12 calculates the square sum of pixel values in a part or all of reference blocks in the reference area in each reference block and stores the calculated value in a storage part 15. An addition part 16 mutually adds the square sum of the current block, the (-2) multiplied value of the inner product between the corrent block corresponding to a moving vector to be retrieved and the reference block, and the square sum of the reference block to find out an error power value. A moving vector determination part 17 detects a minimum value of error power and determines an optimum moving vector.

(19)日本国特許庁（ＪＰ）　(12) **公開特許公報（A）**　(11)特許出願公開番号

特開平11－88891

(43)公開日　平成11年(1999)３月30日

(51)Int.Cl.[6]	識別記号	ＦＩ	
Ｈ０４Ｎ　7/32		Ｈ０４Ｎ　7/137	Ｚ

審査請求　有　　請求項の数２　ＯＬ　（全 22 頁）

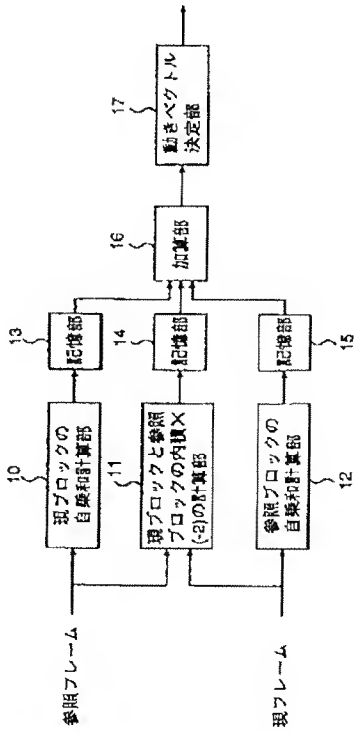
(21)出願番号　特願平10－194893
(62)分割の表示　特願平7－273032の分割
(22)出願日　平成７年(1995)10月20日

(31)優先権主張番号　特願平7－195998
(32)優先日　平７(1995)７月７日
(33)優先権主張国　日本（ＪＰ）

(71)出願人　000004237
日本電気株式会社
東京都港区芝五丁目７番１号
(72)発明者　内藤　幸宏
東京都港区芝五丁目７番１号　日本電気株式会社内
(72)発明者　宮崎　孝
東京都港区芝五丁目７番１号　日本電気株式会社内
(72)発明者　黒田　一朗
東京都港区芝五丁目７番１号　日本電気株式会社内
(74)代理人　弁理士　京本　直樹　（外２名）

(54)【発明の名称】　動き検出方式

(57)【要約】
【課題】　算術論理演算器と積和演算器を備えたプロセッサを用いて動画像の動きベクトルを検出するブロックマッチング型動き検出方式に関し、演算量を大幅に削減することを目的とする。
【解決手段】　１０にて計算された現ブロックの画素値の自乗和を１３に記憶する。１１にて、現ブロックと参照領域内の一部、あるいは全ての参照ブロックとの内積を－２倍した値を参照ブロック毎に計算し、１４に記憶する。１２にて、参照領域内の一部、あるいは全ての参照ブロックの画素値の自乗和を参照ブロック毎に計算し、１５に記憶する。現ブロックの自乗和と、探索する動きベクトルに対応する現ブロックと参照ブロックの内積を－２倍した値と、参照ブロックの自乗和を１６にて加算し誤差電力値とする。１７にて、誤差電力の最小値を検出し、最適な動きベクトルを決定する。



1

【特許請求の範囲】

【請求項１】フレームをいくつかのブロックに分割し、現フレーム中の現ブロックと最も相関の高いブロックを参照フレーム中から検出することによって動きベクトルを検出する動き検出方式において、前記現ブロックの画素値の自乗和を計算する手段と、前記現ブロックと前記参照フレームの参照領域内の一部、あるいは全ての参照ブロックとの内積を－２倍した値を該参照ブロック毎に計算する内積計算手段と、前記参照フレームの参照領域内の一部、あるいは全ての参照ブロックの画素値の自乗和を該参照ブロック毎に計算する第１の参照ブロック自乗和演算手段と、現ブロックの画素値の自乗和と、現ブロックと参照ブロックとの内積を－２倍した値と、参照ブロックの画素値の自乗和とを加算した誤差電力値を順次計算する手段と、前記順次計算された誤差電力値のうち最小となるものを検出し、誤差電力値が最小となったときの参照ブロックに対応する動きベクトルを前記現ブロックの動きベクトルとして出力する手段とを備える動き検出方式において、前記第１の参照ブロック自乗和演算手段に代えて、前記参照フレーム内の全ての参照ブロックの画素値の自乗和を該参照ブロック毎に計算する第２の参照ブロック自乗和演算手段と、該参照ブロック自乗和演算結果を記憶する記憶手段を備え、複数の異なる現ブロックの動き検出において同一の参照ブロックの画素値の自乗和として、前記記憶手段に記憶された参照ブロック自乗和演算結果を再利用することを特徴とする動き検出方式。

【請求項２】フレームをいくつかのブロックに分割し、現フレーム中の現ブロックと最も相関の高いブロックを参照フレーム中から検出することによって動きベクトルを検出する動き検出方式において、前記現ブロックと前記参照フレームの参照領域内の一部、あるいは全ての参照ブロックとの内積を－２倍した値を該参照ブロック毎に計算する内積計算手段と、前記参照フレームの参照領域内の一部、あるいは全ての参照ブロックの画素値の自乗和を該参照ブロック毎に計算する第１の参照ブロック自乗和演算手段と、現ブロックと参照ブロックとの内積を－２倍した値と、参照ブロックの画素値の自乗和とを加算した誤差電力評価値を順次計算する手段と、前記順次計算された誤差電力評価値のうち最小となるものを検出し、誤差電力評価値が最小となったときの参照ブロックに対応する動きベクトルを前記現ブロックの動きベクトルとして出力する手段とを備える動き検出方式において、前記第１の参照ブロック自乗和演算手段に代えて、前記参照フレーム内の全ての参照ブロックの画素値の自乗和を該参照ブロック毎に計算する第２の参照ブロック自乗和演算手段と、該参照ブロック自乗和演算結果を記憶する記憶手段を備え、複数の異なる現ブロックの動き検出において同一の参照ブロックの画素値の自乗和として、前記記憶手段に記憶された参照ブロック自乗和演算結果を再利用することを特徴とする動き検出方式。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、動き検出方式に関し、特に、現フレーム中の現ブロックと最も相関の高いブロックを参照フレーム中から検出することによって動きベクトルを検出する動き検出方式に関する。

【０００２】

【従来の技術】動画像の圧縮符号化においては、動き補償と直交変換を組み合わせた動画像圧縮符号化方式が多く用いられ、動き補償において、画像の動きを検出するために、一般にブロックマッチング型動き検出方式が用いられる。従来のブロックマッチング型動き検出方式としては、例えば特開平１－２９５３７９号公報に示されたものが知られている。図１３は従来の動き検出方式を説明するものであって、（ａ）は現フレームを示し、（ｂ）は参照フレームを示している。フレームをいくつかのブロックに分割し、現フレーム中の現ブロックと最も相関の高いブロックを参照フレーム中から検出することによって動きベクトルを決定する。動画像において連続するフレーム間では動きは小さいと考えられるために、図１３のように参照フレーム中で、相関の高いブロックを探索する範囲を参照領域として設定し、この領域内の全ての参照ブロック、或は意図的に選択された参照ブロックに対して相関を計算する。意図的に参照ブロックを選択する手法としては、１９９３年発行の動画像符号化規格であるＩＳＯ／ＩＥＣ１１１７２－２の８２頁に記載されているロガリズミックサーチ法がある。ロガリズミックサーチ法は、例えば、動きベクトルが（０，０）とその周辺±４の８個の参照ブロックを探索し、その中で最適参照ブロックを決定し、次に、決定された最適参照ブロックの周辺±２の８個の参照ブロックを探索するといった方法で探索していくものである。現フレームの全てのブロックに対して、以上のような探索処理を実行し動きベクトルを検出する。探索の際の評価量である相関は、一般に、現ブロックと参照ブロックの画素値の差分の絶対値、或は画素値の差分の自乗値をブロック全体に対して加算することによって評価される。

【０００３】図１４に算術論理演算器と積和演算器を有するプロセッサの例を示す。相関を評価する際に、現ブロックと参照ブロックの画素値の差分の絶対値和を用いる場合には、現ブロックと参照ブロックの画素値に対して、算術論理演算器において差分演算と絶対値演算と加算演算を実行するため、１画素あたり３サイクルの演算が必要となる。また、評価する際に、現ブロックと参照ブロックの画素値の差分の自乗和を用いる場合には、算術論理演算器で差分演算を、積和演算器で自乗和演算をするため、１画素あたり２サイクルの演算で実行することができる。図１５に相関の評価に自乗和を用いたときのフローチャートを示す。以後、現ブロックと参照ブロ

2

ックの画素値の差分の自乗和を誤差電力値と呼ぶ。ブロックの大きさをＮ×Ｎ個、現ブロックの画素値を$x_{i,j}$、参照領域の画素値を$y_{i,j}$とし、動きベクトルが（ｕ，ｖ）のときの誤差電力値は次式で示される。ただ＊し、Ｎは１より大きい整数であり、$x_{0,0}$を現ブロックの左上の画素値とする。

【０００４】

【数１】

$$e_{u,v} = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \left(x_{i,j} - y_{i+u,j+v}\right)^2 \qquad (1)$$

【０００５】積和演算に必要な時間を１サイクル、差分演算に必要な時間を１サイクルとすると、１ブロック分の誤差電力の計算には、２×Ｎ×Ｎサイクル必要である。この計算を相関を計算する全ての参照ブロックに対して行うことで、１つのブロックの動きベクトルが検出される。この手順を現フレーム内の全ての選択されたブロックに対して順次行う。

【０００６】

【発明が解決しようとする課題】図１５に示すように算術論理演算器と積和演算器を備えたプロセッサで動き検出をする場合、現ブロックと１つの参照ブロック間の相関の計算に１画素当たり２サイクル必要であり、これをブロック内全ての画素に対して行う必要がある。１つの現ブロックに対して複数の参照ブロックとの相関を計算する必要があり、この処理を１つの現フレーム内の全てのブロックに対して実行しなければならないため、膨大な演算量となる。一方、ビデオ会議等で用いられる画像圧縮符号化には実時間処理が要求される。画像圧縮符号化処理においては、動き検出部分の演算量が大きな比重を占めており、装置規模を小さくする、或は性能の向上を図るためには、動き検出部分の演算量をできうる限り削減することが要求される。そこで、本発明の目的は、算術論理演算器と積和演算器を備えたプロセッサで動き検出をする際に必要となる演算量を大幅に削減することにある。

【０００７】

【課題を解決するための手段】第１の発明による動き検出方式は、フレームをいくつかのブロックに分割し、現フレーム中の現ブロックと最も相関の高いブロックを参照フレーム中から検出することによって動きベクトルを検出する動き検出方式において、前記現ブロックの画素値の自乗和を計算する手段と、前記現ブロックと前記参照フレームの参照領域内の一部、あるいは全ての参照ブロックとの内積を－２倍した値を該参照ブロック毎に計算する内積計算手段と、前記参照フレームの参照領域内の一部、あるいは全ての参照ブロックの画素値の自乗和を該参照ブロック毎に計算する第１の参照ブロック自乗和演算手段と、現ブロックの画素値の自乗和と、現ブロックと参照ブロックとの内積を－２倍した値と、参照ブロックの画素値の自乗和とを加算した誤差電力値を順次計算する手段と、前記順次計算された誤差電力値のうち最小となるものを検出し、誤差電力値が最小となったときの参照ブロックに対応する動きベクトルを前記現ブロックの動きベクトルとして出力する手段とを備えていることを特徴とする。

【０００８】第２の発明による動き検出方式は、フレームをいくつかのブロックに分割し、現フレーム中の現ブロックと最も相関の高いブロックを参照フレーム中から検出することによって動きベクトルを検出する動き検出方式において、前記現ブロックと前記参照フレームの参照領域内の一部、あるいは全ての参照ブロックとの内積を－２倍した値を該参照ブロック毎に計算する内積計算手段と、前記参照フレームの参照領域内の一部、あるいは全ての参照ブロックの画素値の自乗和を該参照ブロック毎に計算する第１の参照ブロック自乗和演算手段と、現ブロックと参照ブロックとの内積を－２倍した値と、参照ブロックの画素値の自乗和とを加算した誤差電力評価値を順次計算する手段と、前記順次計算された誤差電力評価値のうち最小となるものを検出し、誤差電力評価値が最小となったときの参照ブロックに対応する動きベクトルを前記現ブロックの動きベクトルとして出力する手段とを備えていることを特徴とする。

【０００９】第３の発明による動き検出方式（請求項に係る発明）は、第１、あるいは、第２の発明において、前記第１の参照ブロックの自乗和演算手段に代えて、前記参照フレーム内の全ての参照ブロックの画素値の自乗和を該参照ブロック毎に計算する第２の参照ブロック自乗和演算手段と、該参照ブロック自乗和演算結果を記憶する記憶手段を備え、複数の異なる現ブロックの動き検出において同一の参照ブロックの画素値の自乗和として、前記記憶手段に記憶された参照ブロック自乗和演算結果を再利用することを特徴とする。

【００１０】第４の発明による動き検出方式は、第１、第２、あるいは、第３の発明において、前記第１または第２の参照ブロック自乗和演算手段が、参照ブロックの水平方向または垂直方向のブロック毎の自乗和からブロック全体の自乗和の計算をする手段とで構成されることを特徴とする。

【００１１】第５の発明による動き検出方式は、第１、第２、第３、あるいは、第４の発明において、前記内積計算手段が、現ブロックの画素値を水平方向、垂直方向共に２で間引いた４種類のブロックから加減算により計算される計９個の副次現ブロックを出力とする現ブロック前処理手段と、参照領域の画素値を水平方向、垂直方

向共に２で間引いた４種類の領域から加減算により計算される計９個の副次参照領域を出力とする参照領域前処理手段と、前記副次現ブロックと前記副次参照領域の内積を計算する９個の副次内積計算手段と、前記９個の副次内積計算手段の出力から、加減算により内積結果を計算する後処理手段とで構成されることを特徴とする。

【００１２】第６の発明による動き検出方式は、第５の発明において、前記副次内積計算手段として、第５の発明に記載の内積計算手段を再帰的に用いることを特徴とする。

【００１３】第７の発明による動き検出方式は、フレームをいくつかのブロックに分割し、現フレーム中の現ブロックと最も相関の高いブロックを参照フレーム中から検出する際に、前記参照フレームの参照領域内で、基準参照ブロックと基準参照ブロックの周辺に位置する参照ブロックを候補として最適な参照ブロックを決定し、これを基準参照ブロックとする操作を繰り返して動きベクトルを検出する動き検出方式において、前記基準参照ブロックの画素値の自乗和と前記周辺の参照ブロックの画素値の自乗和との差分をそれぞれ計算する手段と、現ブロックと基準参照ブロックの内積を－２倍した値を計算する手段と、現ブロックと参照ブロックの内積を２倍した値を計算する手段と、前記計算結果と、前記記憶手段に記憶された基準参照ブロックの画素値の自乗和と当該参照ブロックの画素値の自乗和との差分値と、現ブロックと基準参照ブロックの内積を－２倍した値を加算する手段と、各探索参照ブロックについて順次計算された前記加算結果から値が最小となるものを検出し、前記最小＊値が負の値のときは前記最小値となったときの参照ブロックに対応する動きベクトルを前記現ブロックの動きベクトルとして出力し、前記最小値が負の値でないときは基準参照ブロックに対応する動きベクトルを前記現ブロックの動きベクトルとして出力する手段を備えていることを特徴とする。

【００１４】第８の発明による動き検出方式は、フレームをいくつかのブロックに分割し、現フレーム中の現ブロックと最も相関の高いブロックを参照フレーム中から検出することによって動きベクトルを検出する動き検出方式において、前記現ブロックの画素値の自乗和を計算し、前記現ブロックと前記参照フレームの参照領域内の一部、あるいは全ての参照ブロックとの内積を－２倍した値を該参照ブロック毎に計算し、前記参照フレームの参照領域内の一部、あるいは全ての参照ブロックの画素値の自乗和を該参照ブロック毎に計算し、現ブロックの画素値の自乗和と、現ブロックと参照ブロックとの内積を－２倍した値と、参照ブロックの画素値の自乗和とを加算した誤差電力値を順次計算し、前記順次計算された誤差電力値のうち最小となるものを検出し、誤差電力値が最小となったときの参照ブロックに対応する動きベクトルを前記現ブロックの動きベクトルとして出力することを特徴とする。

【００１５】

【作用】第１の発明による動き検出方式の作用を説明する。式（１）を次式のように展開する。

【００１６】

【数２】

$$e_{u,v}=\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}x_{i,j}^{2}-2\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}x_{i,j}y_{i+u,j+v}+\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}y_{i+u,j+v}^{2}\qquad(2)$$

【００１７】式（１）をこのように展開することにより、各項の演算量を減少させたり、各項のデータを他に有効に用いたりすることができる。以下、この点について詳述する。

【００１８】上式の第１項は、現ブロックの画素値の自乗和である。この値は参照ブロックの位置を表す（ｕ，ｖ）に係わらず一定であるので、１つの現ブロックの動きベクトルを検出する際に１回だけ計算すればよい。第２項は、現ブロックと参照ブロックの内積結果を－２倍したものである。この計算には１ブロックあたりＮ×Ｎ＋１サイクル必要である。この計算は探索する参照ブロック全てに対して行う必要があり、式（１）の場合の約半分の演算量である。第３項は、参照ブロックの画素値の自乗和であり、１ブロックあたりＮ×Ｎサイクル必要である。この計算も第２項と同様に探索する参照ブロック全てについて計算する必要がある。その結果、第２項と第３項の計算で式（１）と等しいサイクル数が必要となり、これに第１項の計算が必要となるので、演算量の削減にはならない。

【００１９】しかしながら、探索する各参照ブロックについて第３項の自乗和の計算を行う場合、重複する画素が多く存在する。図８にこの様子を示す。簡単化のため、ブロックサイズを４×４、動きベクトルの探索範囲を－２～＋２としている。この場合、参照領域の大きさは８×８となる。図８には、例として動きベクトルが（０，０）の場合と、（１，－１）の場合の参照ブロックの様子が示されている。この場合、ブロック内の１６個の画素のうち９個の画素が２つの参照ブロック間で重

複している。つまり、１つの動きベクトルを検出する際に、探索する全ての参照ブロックの画素の自乗和を予め計算する必要があるが、その際に重複する画素の自乗演算を省くことにより、第３項の計算に必要なサイクル数を大幅に減らすことができる。これにより、１つの現ブロックあたりに必要となる計算量を減らすことができ結果として、計算に必要なサイクル数を減らすことができる。

【００２０】また式（２）の第２項は、２次元畳み込み演算であり、ＦＦＴ（高速フーリエ変換）や第５の発明で示した方法を用いることにより、その計算量を減らすことが可能である。

【００２１】さらに、式（２）の第１項は、現ブロックの画素値の自乗和であって、ＣＣＩＴＴの勧告H．２６１等の画像の圧縮符号化においては、動き検出後に実行されるＩＮＴＲＡ／ＩＮＴＥＲの予測モード判定時に用いることができる値である。

【００２２】第２の発明による動き検出方式の作用を説明する。第１の発明と同様に、式（１）を式（２）のように展開して計算する。誤差電力値である式（２）を、探索する全ての（ｕ，ｖ）について計算し、最小値を示す（ｕ，ｖ）を検出する必要がある。しかし、式（２）の第１項は（ｕ，ｖ）に関わらず一定値であるので、式（２）の大小比較をする際には不必要な項である。第１の発明では第１項を１回だけ計算し、この値を再利用していたが、第２の発明では第１項を計算せずに、第２項と第３項の和を誤差電力評価値として最小値検出の際の＊比較に用いる。このようにすることにより、第１の発明の作用である、第１項の値をＩＮＴＲＡ／ＩＮＴＥＲの予測モード判定時に再利用できるという特徴と、誤差電力の最小値を求めることができるという特徴は消滅するが、第１項の計算をする必要がないため、演算量をさらに削減することができる。

【００２３】第３の発明による動き検出方式の作用を説明する。第３の発明では現フレーム内の各ブロックに対応する参照領域にも重複する部分があることを利用する。つまり、異なる参照領域中の参照ブロックの間にも重複する画素が存在する。例えば図９にこのような例を示す。隣あったブロックをブロック１とブロック２とする。これらの参照領域は図９に示すように重複している。例えば、ブロック１に関する動きベクトルが（２，－２）の参照ブロックと、ブロック２に関する動きベクトルが（－２，－２）の参照ブロックは全く同一になる。よって、１フレーム分の動き検出処理をする際に必要な第３項の値を、予め計算しておくことで、第３項の計算に必要なサイクル数を更に減らすことができる。

【００２４】第４の発明による動き検出方式の作用を説明する。第１、第２、第３の発明同様に式（１）を式（２）のように展開して動き検出を行う。本発明は、式（２）の第３項を参照領域の水平方向と垂直方向で分離して計算することにより演算量を削減するものである。

【００２５】式（２）の第３項を$f_{u,v}$　とする。

【００２６】

【数３】

$$f_{u,v}=\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}y_{i+u,j+v}^{2} \qquad (3)$$

【００２７】例えば、探索する動きベクトルを

【００２８】

【数４】

$$u,v=-\frac{M}{2}+1\sim\frac{M}{2}$$

【００２９】とし、この参照領域内の全ての参照ブロックを探索する場合を考える。ただし、Ｍは偶数とする。この場合、式（３）を

【００３０】

【数５】

$$u,v=-\frac{M}{2}+1\sim\frac{M}{2}$$

【００３１】について計算することになる。

【００３２】第１の発明の作用でも述べたように、参照領域中の参照ブロック間には重複する画素が多く存在する。この重複する画素の自乗計算を省くことにより、第３項に必要な演算量を大幅に削減することができる。

【００３３】本発明では、この第３項の計算を水平方向、垂直方向に分離して行う。例えば水平方向、垂直方向の順に計算する場合について説明する。まず、参照ブロックの水平方向の自乗和を以下のように計算する。

【００３４】

【数６】

9　　10

$$f'_{k,l} = \sum_{i=0}^{N-1} y^2_{i+k,l},$$

$$k = -\frac{M}{2}+1,\cdots,\frac{M}{2},\quad l = -\frac{M}{2}+1,\cdots,N+\frac{M}{2}-1 \qquad (4)$$

【００３５】次に、これらの結果を垂直方向に加算する　10＊【００３６】
ことによって、第３項を求める。つまり、　　＊　　【数７】

$$f_{u,v} = \sum_{j=0}^{N-1} f'_{u,j+v},$$

$$u,v = -\frac{M}{2}+1,\cdots,\frac{M}{2} \qquad (5)$$

【００３７】ここで、式（４）と式（５）は次のように　20※【００３８】
巡回的に高速に計算できる。　　※　　【数８】

$$f'_{k,l} = f'_{k-1,l} + y^2_{k+N-1,l} - y^2_{k-1,l} \qquad (6)$$

$$f_{u,v} = f_{u,v-1} + f'_{u,v+N-1} - f'_{u,v-1} \qquad (7)$$

【００３９】この方法によれば、第３項の必要演算量を$N^2 \times M^2$から（N＋M－１）×（N＋２M）＋M×（N＋２M）に削減できる。例えば、N＝１６，M＝３２とすると、演算量は２６２，１４４サイクルから６，３２０サイクルに削減されることになる。なお、式（２）の第３項を水平方向と垂直方向に分離して計算する本発明は、Mが偶数、奇数によらずに適用できるものである。

★【００４０】第５の発明による動き検出方式の作用を説明する。第１、第２、第３、第４の発明同様に式（１）を式（２）のように展開して動き検出を行う。この際、式（２）の第２項の計算を以下のように行う。
【００４１】$c_{u,v}$　を以下のように定義する。
【００４２】
【数９】

★

$$c_{u,v} = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} x_{i,j}\,y_{i+u,j+v} \qquad (8)$$

【００４３】この$c_{u,v}$　を
【００４４】
【数１０】

$$u,v = -\frac{M}{2}+1 \sim \frac{M}{2}$$

【００４５】について計算し、－２倍することが、動きベクトルの検出範囲を
【００４６】
【数１１】

$$u,v = -\frac{M}{2}+1 \sim \frac{M}{2}$$

【００４７】としたときの参照領域中の全ての参照ブロックに関する式（２）の第２項を計算することになる。ただし、Mは偶数とする。
【００４８】
【外１】

$$\hat{x}_{i,j},\ \hat{y}_{i,j}$$

50　【００４９】を以下のように定義する。

【００５０】　　＊　＊【数１２】

$$\hat{x}_{i,j} = [x_{i,j}, x_{i+2,j}, \cdots, x_{i+N-2,j}] \quad (9)$$

$$\hat{y}_{i,j} = [y_{i,j}, y_{i+2,j}, \cdots, y_{i+N-2,j}] \quad (10)$$

【００５１】これらを用いて、$X_{i,j}$　，$Y_{i,j}$　を以下　※【００５２】
のように定義する。　　※　【数１３】

$$X_{i,j} = [\hat{x}_{i,j}, \hat{x}_{i,j+2}, \cdots, \hat{x}_{i,j+N-2}]^T \quad (11)$$

$$Y_{i,j} = [\hat{y}_{i,j}, \hat{y}_{i,j+2}, \cdots, \hat{y}_{i,j+N-2}] \quad (12)$$

【００５３】$X_{i,j}$　，$Y_{i,j}$　は、それぞれN^2　／４　★【００５４】これらベクトルを用いると式（８）のｃ
個の要素を持つベクトルである。各要素は、それぞれ現　$_{u,v}$　、$C_{u+1,v}$　、$C_{u,v+1}$　、$C_{u+1,v+1}$　は以下のよ
ブロック、参照領域の（ｉ，ｊ）位置のＮ×Ｎのブロッ　うに表される。
クの画素値を水平方向、垂直方向共に２で間引いたもの　【００５５】
である。　　★20　【数１４】

$$C_{u,v} = \begin{bmatrix} c_{u,v} \\ c_{u+1,v} \\ c_{u,v+1} \\ c_{u+1,v+1} \end{bmatrix}$$

$$= \begin{bmatrix} Y_{u,v} & Y_{u+1,v} & Y_{u,v+1} & Y_{u+1,v+1} \\ Y_{u+1,v} & Y_{u+2,v} & Y_{u+1,v+1} & Y_{u+2,v+1} \\ Y_{u,v+1} & Y_{u+1,v+1} & Y_{u,v+2} & Y_{u+1,v+2} \\ Y_{u+1,v+1} & Y_{u+2,v+1} & Y_{u+1,v+2} & Y_{u+2,v+2} \end{bmatrix} \begin{bmatrix} X_{0,0} \\ X_{1,0} \\ X_{0,1} \\ X_{1,1} \end{bmatrix} \quad (13)$$

【００５６】式（１３）中には１６個のＹとＸの内積演　【００５７】
算が存在するが、次式の変換式を用いるとこの内積演　【数１５】
の個数を削減することができる。

13 14

$$\begin{bmatrix} A & B \\ B & C \end{bmatrix}\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} (A+B)X - B(X-Y) \\ (B+C)Y + B(X-Y) \end{bmatrix} \tag{14}$$

【００５８】式（１３）中の４箇所に対して式（１４）の変形を施し、まとめると$C_{u,v}$は以下のように表される。

＊【００５９】【数１６】＊

$$\begin{aligned} C_{u,v} = & \begin{bmatrix} (Y_{u,v}+Y_{u+1,v})X_{0,0} - Y_{u+1,v}(X_{0,0}-X_{1,0}) \\ (Y_{u+1,v}+Y_{u+2,v})X_{1,0} + Y_{u+1,v}(X_{0,0}-X_{1,0}) \\ (Y_{u,v+1}+Y_{u+1,v+1})X_{0,0} - Y_{u+1,v+1}(X_{0,0}-X_{1,0}) \\ (Y_{u+1,v+1}+Y_{u+2,v+1})X_{1,0} + Y_{u+1,v+1}(X_{0,0}-X_{1,0}) \end{bmatrix} \\ & + \begin{bmatrix} (Y_{u,v+1}+Y_{u+1,v+1})X_{0,1} - Y_{u+1,v+1}(X_{0,1}-X_{1,1}) \\ (Y_{u+1,v+1}+Y_{u+2,v+1})X_{1,1} + Y_{u+1,v+1}(X_{0,1}-X_{1,1}) \\ (Y_{u,v+2}+Y_{u+1,v+2})X_{0,1} - Y_{u+1,v+2}(X_{0,1}-X_{1,1}) \\ (Y_{u+1,v+2}+Y_{u+2,v+2})X_{1,1} + Y_{u+1,v+2}(X_{0,1}-X_{1,1}) \end{bmatrix} \\ = & \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} Y_{u,v}+Y_{u+1,v} & Y_{u,v+1}+Y_{u+1,v+1} \\ Y_{u,v+1}+Y_{u+1,v+1} & Y_{u,v+2}+Y_{u+1,v+2} \end{bmatrix}\begin{bmatrix} X_{0,0} \\ X_{0,1} \end{bmatrix} \\ & + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} Y_{u+1,v}+Y_{u+2,v} & Y_{u+1,v+1}+Y_{u+2,v+1} \\ Y_{u+1,v+1}+Y_{u+2,v+1} & Y_{u+1,v+2}+Y_{u+2,v+2} \end{bmatrix}\begin{bmatrix} X_{1,0} \\ X_{1,1} \end{bmatrix} \\ & + \begin{bmatrix} -1 & 0 \\ 1 & 0 \\ 0 & -1 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} Y_{u+1,v} & Y_{u+1,v+1} \\ Y_{u+1,v+1} & Y_{u+1,v+2} \end{bmatrix}\begin{bmatrix} X_{0,0}-X_{1,0} \\ X_{0,1}-X_{1,1} \end{bmatrix} \end{aligned} \tag{15}$$

【００６０】次に、式（１５）中の３箇所に対して式（１４）の変形を施し、まとめると以下のようになる。

【００６１】【数１７】

15 16

$$C_{u,v}=\begin{bmatrix}(Y_{u,v}+Y_{u+1,v}+Y_{u,v+1}+Y_{u+1,v+1})X_{0,0}\\(Y_{u+1,v}+Y_{u+2,v}+Y_{u+1,v+1}+Y_{u+2,v+1})X_{1,0}\\(Y_{u,v+1}+Y_{u+1,v+1}+Y_{u,v+2}+Y_{u+1,v+2})X_{0,1}\\(Y_{u+1,v+1}+Y_{u+2,v+1}+Y_{u+1,v+2}+Y_{u+2,v+2})X_{1,1}\end{bmatrix}$$

$$+\begin{bmatrix}-1&0\\0&-1\\1&0\\0&1\end{bmatrix}\begin{bmatrix}(Y_{u,v+1}+Y_{u+1,v+1})(X_{0,0}-X_{0,1})\\(Y_{u+1,v+1}+Y_{u+2,v+1})(X_{1,0}-X_{1,1})\end{bmatrix}$$

$$+\begin{bmatrix}-1&0\\1&0\\0&-1\\0&1\end{bmatrix}\begin{bmatrix}(Y_{u+1,v}+Y_{u+1,v+1})(X_{0,0}-X_{1,0})\\(Y_{u+1,v+1}+Y_{u+1,v+2})(X_{0,1}-X_{1,1})\end{bmatrix}$$

$$+\begin{bmatrix}1\\-1\\-1\\1\end{bmatrix}Y_{u+1,v+1}(X_{0,0}-X_{1,0}-X_{0,1}+X_{1,1}) \qquad (16)$$

【００６２】$C_{u,v}$、$C_{u-1,v}$、$C_{u,v-1}$、$C_{u-1,v-1}$を同時に計算し、さらに式（１４）を利用して変形することにより、式（１３）の１６個の内積演算が、式（１６）の９個の副次内積演算に削減される。ブロックサイズがＮ×Ｎ、探索点数がＭ×Ｍの場合、式（８）、あるいは式（１３）を直接計算すると$N^2\times M^2$サイクル必要である。式（１６）のように変形することによって、副次内積演算の演算量は合計で９／１６×（$N^2\times M^2$）サイクルに削減される。これに加えて、式（１６）中の$X_{0,0}-X_{0,1}$、$X_{1,0}-X_{1,1}$、$X_{0,0}-X_{1,0}$、$X_{0,1}-X_{1,1}$、$X_{0,0}-X_{1,0}-X_{0,1}+X_{1,1}$の現ブロックの前処理に$5N^2$／４サイクル、$Y_{u,v}+Y_{u+1,v}+Y_{u,v+1}+Y_{u+1,v+1}$、$Y_{u-1,v}+Y_{u,v}+Y_{u-1,v+1}+Y_{u,v+1}$、$Y_{u,v-1}+Y_{u+1,v-1}+Y_{u,v}+Y_{u+1,v}$、$Y_{u-1,v-1}+Y_{u,v-1}+Y_{u-1,v}+Y_{u,v}$、$Y_{u-1,v}+Y_{u+1,v}+Y_{u-1,v+1}+Y_{u+1,v+1}$の参照領域の前処理に２（Ｎ＋Ｍ－２）×（Ｎ＋Ｍ－１）

17

＋（Ｎ＋Ｍ－２）^{2}　サイクル、各副次内積演算結果から式（１６）に従い$C_{u,v}$　を計算する後処理に１０Ｍ^2／４サイクル必要となる。Ｎ＝１６、Ｍ＝３２の場合、副次内積演算部が１４７×１０^3　サイクルに対し、前処理、後処理部は合計で９．３２×１０^3　サイクルであり、これらのオーバーヘッドは十分小さいものとなる。

【００６３】図１０に本発明に基づく式（２）の第３項の内積演算部を示す。式（８）の内積演算が、式（１６）のように９個の副次内積演算に分割される。現ブロックの前処理にて、現ブロックの画素値から式（１６）中のＸで表される副次現ブロックを計算する。分割された副次内積演算部に対し、参照領域の前処理にて参照領域画素値から生成された式（１６）中のＹで表される副次参照領域値が入力される。各副次内積演算部の出力値から式（１６）に従って、後処理にて$C_{u,v}$　が生成される。

【００６４】第６の発明による動き検出方式の作用を説明する。第６の発明では第５の発明において分割された副次内積演算を、式（１６）を用いて、さらに、再帰的に分割する。例えば、第５の発明において分割された９個の副次内積演算全てに対して、さらに分割を行うと、計８１個の副次内積演算となる。９個の副次内積演算に分割した場合の、副次内積演算部の演算量は、９／１６＊

18

＊＋（Ｎ^2　×Ｍ^2　）サイクルであったが、８１個に分割した場合、演算量は、（９／１６）^{2}　×（Ｎ^2×Ｍ^2　）サイクルに削減される。

【００６５】第７の発明による動き検出方式の作用を説明する。本発明は、ロガリズミックサーチ法に見られるような、ある基準となる参照ブロックの周辺の参照ブロックを探索する場合の演算量を削減できるものである。以後、この基準となる参照ブロックを基準参照ブロックと呼ぶ。例えば、基準参照ブロック（ｕ０，ｖ０）の周辺の参照ブロックである（ｕ０＋１，ｖ０）、（ｕ０＋１，ｖ０＋１）、（ｕ０，ｖ０＋１）、（ｕ０－１，ｖ０＋１）、（ｕ０－１，ｖ０）、（ｕ０－１，ｖ０－１）、（ｕ０，ｖ０－１）、（ｕ０＋１，ｖ０－１）との相関を計算する場合を考える。この場合、相関を計算する参照ブロックは８個で、基準参照ブロックを含めた９個の参照ブロックから最適なブロックを検出する。この場合、（ｕ０，ｖ０）に関する式（２）と他の８個の周辺ベクトルに関する式（２）の中から最小値を検出する必要がある。本発明では、基準ベクトルである（ｕ０，ｖ０）に関する式（２）と周辺ベクトルに関する式（２）との差分を用いて、最小値を示すベクトルを検出する。例えば$e_{u0,v0}$　と$e_{u0+1,v0-1}$　の差分は、

【００６６】

【数１８】

$$
\begin{aligned}
e_{u0,v0} - e_{u0+1,v0-1} &= -2\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} x_{i,j}y_{u0+i,v0+j} \\
&\quad +2\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} x_{i,j}y_{u0+i+1,v0+j-1} \\
&\quad +\left\{\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} y_{u0+i,v0+j}^2 - \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} y_{u0+i+1,v0+j-1}^2\right\} \qquad (17)
\end{aligned}
$$

【００６７】となる。上式の第１項は現ブロックと基準参照ブロックの内積であり、１回だけ計算する必要がある。第２項は現ブロックと参照ブロックとの内積であり、これに関しては探索する全ての参照ブロックについて計算する。第３項は、基準参照ブロックの画素の自乗和と探索する参照ブロックの画素の自乗和の差分である。

【００６８】基準参照ブロックと周辺の参照ブロックとの間には共通の画素が存在し、差分を取ることからこの共通部分の計算は不要となる。ブロックサイズが４×４のときの様子を図１１に示す。基準参照ブロックと基準参照ブロックからみた動きベクトルが（１，－１）の参照ブロックとの間には９個の共通画素が存在するので、この部分の自乗値を計算する必要はない。動きベクトルを水平方向、垂直方向共に±１の範囲で探索する場合、図１１のように、４点の画素について自乗値を計算する必要がなくなる。この例ではブロックサイズが４×４であるが、ブロックサイズが大きくなれば、自乗値を計算する必要のない画素はさらに増加する。例えばブロックサイズが１６×１６で基準参照ブロックの周囲±１を探索する場合１８×１８点中自乗値を計算する必要のない点は１４×１４点となる。

【００６９】このようにして、基準参照ブロックとその周辺の参照ブロックの中から最適な参照ブロックを少ない演算量で検出できる。検出した動きベクトルを（Ｕ，Ｖ）とすれば、この動きベクトルに対応する参照ブロッ

19

クと現ブロックの差分の自乗和e_{MV}　は式（１７）から計算することができる。

【００７０】

【発明の実施の形態】以下、本発明の実施例について図面を参照しながら説明する。図１は第１、第３の発明の実施例を示すブロック図である。現ブロックの自乗和計算部１０にて、現ブロックの画素の自乗和、つまり、式（２）の第１項を計算し、記憶部１３に記憶する。内積計算部１１にて、現ブロックと参照ブロックの内積値を－２倍したもの、つまり、式（２）の第２項を全ての参照ブロック、あるいは、一部の参照ブロックについて計算し、記憶部１４に記憶する。参照ブロックの自乗和計算部１２にて、参照ブロック画素の自乗和、つまり、式（２）の第３項を全ての参照ブロック、あるいは、一部の参照ブロックいついて計算し、記憶部１５に記憶する。加算部１６において、記憶部１３に記憶された現ブロックの画素値の自乗和と記憶部１４、記憶部１５に記憶された計算結果の中で、探索中の動きベクトルに対応する結果を加算し式（２）の誤差電力とする。加算部１６で計算された誤差電力を動きベクトル決定部１７で比較し、最小の誤差電力を示す動きベクトルを出力する。

【００７１】内積計算部１１と参照ブロックの自乗和計算部１２では探索する一部、あるいは全ての動きベクトルに対応する値を計算する。一部の動きベクトルに対応する値を計算する場合には、全て計算する場合に比べて、演算量削減の効果は小さくなるが、記憶部１４、記憶部１５の記憶容量を削減することができる。

【００７２】図２は、第２、第３の発明の実施例を示すブロック図である。図１と異なり、現ブロックの自乗和の計算を行わない。このようにすることにより、第１の発明の作用である、第１項の値をＩＮＴＲＡ／ＩＮＴＥＲの予測モード判定時に再利用できるという特徴と、誤差電力の最小値を求めることができるという特徴は消滅するが、第１項の計算をする必要がないため、演算量をさらに削減することができる。

【００７３】以下に各発明の詳細な実施例を示す。図３は第１の発明に基づき、１つの現ブロックの動きベクトルを検出する際の処理過程を示すものである。ステップ３０で、最初に探索する動きベクトルを設定する。また、ステップ３６で用いる最小値クリアしておく。ステップ３１で、現ブロックの画素値の自乗和である式（２）の第１項を計算し記憶する。ステップ３２で、現ブロックと、参照領域内の一部、あるいは全ての参照ブロックとの内積を－２倍した値をそれぞれ計算し記憶する。ステップ３３で、参照領域内の一部、あるいは全ての参照ブロックの画素値の自乗和をそれぞれ計算し記憶する。ステップ３４～ステップ３９で各動きベクトルを探索する。ステップ３４で、現在候補となっている動きベクトルに対応する式（２）の第２、第３項が計算済みかどうかを確認し、計算済みであれば、ステップ３５で

20

これらの値と式（２）の第１項を加算し誤差電力を求める。計算済みでなければ、ステップ３２に戻り、必要な値を計算する。ステップ３２、３３で必要な値を全て計算した場合には、ステップ３４の確認は不要となる。ステップ３６では、計算した誤差電力が探索してきた動きベクトルの中で最小値であったかを判定する。最小値であった場合にはステップ３７で最小値を更新し、このときの動きベクトルを記憶しておく。ステップ３８では、探索する動きベクトルについて全て探索が終了したかを判定し、未了であればステップ３９にて次の動きベクトルを設定してステップ３４に戻る。終了していれば、ステップ４０で、誤差電力の最小値と最小値に対応する動きベクトルを出力する。

【００７４】図４は第２の発明による動き検出方式の処理過程を示す図である。図３の処理過程と違い、式（２）の第１項の計算をせず、ステップ４５で、式（２）の第２項と第３項を加算した値を誤差電力評価値として、ステップ４６で比較している。そのために、ステップ５０では最終結果として動きベクトルのみを出力し、誤差電力の最小値は出力しない。

【００７５】第３の発明は、第１、第２の発明における式（２）の第３項の計算を１個の動きベクトルの検出に必要な値ではなく、現フレーム全てのブロックの動きベクトルの検出に必要な第３項の値の一部、あるいは全ての場合について計算する。図５は第３の発明に基づき、現フレーム全ての動きベクトルを検出する際の処理過程を示す図である。本図は一実施例として第２の発明と第３の発明を組み合わせた場合を示している。ステップ６０で、１フレーム分の動きベクトルの検出に必要な式（２）の第３項を全て計算し記憶する。ステップ６１で、あるブロックの動きベクトルを探索する際の最初に探索する動きベクトルを設定し、ステップ６４で用いる誤差電力評価値の最小値をクリアしておく。ステップ６２で、設定された動きベクトルの検出に必要な式（２）の第２項の全てを計算し記憶する。ステップ６３で、計算済みの式（２）の第２項、第３項を加算し誤差電力評価値を求める。ステップ６４で、この誤差電力評価値が最小値であるかを判定する。最小値であれば、ステップ６５で最小値を更新し、このときの動きベクトルを保存しておく。ステップ６６で、全ての動きベクトル候補の探索が終了したかを判定し、終了していなければステップ６７で次の動きベクトル候補を設定しステップ６３に戻る。終了したならば、ステップ６８で、検出された動きベクトルを出力する。ステップ６９で、１フレーム分の動きベクトルを全て検出したかを判定し、検出したならば終了し、いていなければ、ステップ６１に戻り次のブロックの動きベクトルの検出を行う。

【００７６】第４の発明の一実施例として、式（２）の第３項の演算を水平方向、垂直方向の順に分離して計算する方式を示す。簡単化のためにブロックサイズを４×

21

４、参照領域の大きさを８×８とし、便宜上、参照領域の画素を$y_{2,2}$～$y_{5,5}$とする。この場合、参照領域中に含まれる参照ブロックの数は水平方向５個、垂直方向５個の計２５個となる。この参照ブロックを全て探索する、このため、動きベクトルの範囲を－２～２とする。

【００７９】により計算する。次に垂直方向の演算として、式（５）の$f_{-2,-2}$～$f_{2,2}$を式（７）の再帰式を用いて計算し記憶する。ただし、初期値$f_{u,-2}$は、

$$f_{u,-2}=\sum_{j=-2}^{1} f'_{u,j}, \qquad u=-2,-1,\cdots,2 \tag{19}$$

【００８１】により計算する。このようにして計算した式（２）の第３項を用いて、第１、第２、あるいは第３の発明による動き検出方式で動き検出を行う。例えば、ブロックのサイズを１６×１６、フレームのサイズを１７６×１４４、動き検出の範囲を－７～７としこの範囲内の参照ブロックを全て探索する場合を考える。積和演算、算術演算共に１サイクルで実行するプロセッサ型動き検出装置の場合、式（１）を計算して評価する従来法では、１フレーム分の動きを検出するのに、１１．４×10^6サイクル必要である。本第４の発明と第１の発明を用いた手法により同様の動き検出を実行する場合、１フレーム分の動きの検出が、６．０１×10^6サイクルで実現可能である。これは従来法の５２．７％の演算量に相当する。本第４の発明と第１と第３の発明を組み合わせた手法により同様の動き検出を実行する場合、１フレーム分の動きの検出が、５．８５×10^6サイクル★で実現可能である。これは従来法の５１．３％の演算量に相当する。

22

＊【００７７】まず水平方向の演算として、式（４）の$f'_{2,2}$～$f'_{2,5}$を式（６）の再帰式を用いて計算し記憶する。ただし、初期値$f'_{2,1}$は、

【００７８】

【数１９】

＊

$$f'_{-2,l}=\sum_{i=-2}^{1} y^2_{i,l}, \qquad l=-2,-1,\cdots,5 \tag{18}$$

※【００８０】

【数２０】

※

【００８２】第５の発明の第１の実施例として、式（２）の第２項の演算方法を示す。簡単化のためにブロックサイズを４×４とし、探索する動きベクトルを－２～１までの４×４個とする。この場合の参照領域は、７×７となる。

【００８３】式（２）の第２項の計算は、この場合、式（１６）の$C_{2,-2}$、$C_{0,2}$，$C_{2,0}$、$C_{0,0}$を計算する。それぞれの計算は９個の副次内積演算からなるが、ここでは、式（１６）中の第５番目の副次内積演算である（$Y_{u,v-1}$＋$Y_{u+1,v-1}$）（$X_{0,0}$－$X_{0,1}$）について説明する。まず、式（１６）中の現ブロックの前処理を行う。上記第５番目の内積演算に関する副次現ブロックである$X_{0,0}$－$X_{0,1}$は次のようになる。

【００８４】

$$X_{0,0}-X_{0,1}=[x_{0,0},x_{2,0},x_{0,2},x_{2,2}]^T-[x_{0,1},x_{2,1},x_{0,3},x_{2,3}]^T=[x_{0,0}-x_{0,1},x_{2,0}-x_{2,1},x_{0,2}-x_{0,3},x_{2,2}-x_{2,3}]^T \tag{20}$$

次に、式（１６）中の参照領域の前処理を行う。上記第５番目の内積演算に関する参照領域前処理値である$Y_{u,v-1}$－$Y_{u-1,v-1}$の計算について説明する。

☆（ｕ，ｖ）が、（－２，－２）、（０，－２）、（－２，０）、（０，０）について上式を計算する必要があるが、それぞれの（ｕ，ｖ）について$Y_{u,v-1}$－$Y_{u-1,v-1}$を計算すると以下のようになる。（ｕ，ｖ）＝（－２，－２）のときは、

$$Y_{2,-1}-Y_{1,-1}=[y_{2,-1},y_{3,-1},y_{2,0},y_{3,0}]-[y_{1,-1},y_{1,-1},y_{1,0},y_{1,0}]=[y_{2,-1}-y_{1,-1},y_{3,-1}-y_{1,-1},y_{2,0}-y_{1,0},y_{3,0}-y_{1,0}] \tag{21}$$

（ｕ，ｖ）＝（０，－２）のときは、

$$Y_{0,-1}-Y_{-1,-1}=[y_{0,-1},y_{1,-1},y_{0,0},y_{1,0}]-[y_{-1,-1},y_{-1,-1},y_{-1,0},y_{-1,0}]=[y_{0,-1}-y_{-1,-1},y_{1,-1}-y_{-1,-1},y_{0,0}-y_{-1,0},y_{1,0}-y_{-1,0}] \tag{22}$$

（ｕ，ｖ）＝（－２，０）のときは、

23　　24

$Y_{-2,1} - Y_{-1,1} = [y_{-2,1}, y_{0,1}, y_{-2,3}, y_{0,3}] - [y_{-1,1}, y_{1,1}, y_{-1,3}, y_{1,3}] = [y_{-2,1} - y_{-1,1}, y_{0,1} - y_{1,1}, y_{-2,3} - y_{-1,3}, y_{0,3} - y_{1,3}]$　　（２３）

（ｕ，ｖ）＝（０，０）のときは、

$Y_{0,1} - Y_{1,1} = [y_{0,1}, y_{2,1}, y_{0,3}, y_{2,3}] - [y_{1,1}, y_{3,1}, y_{1,3}, y_{3,3}] = [y_{0,1} - y_{1,1}, y_{2,1} - y_{3,1}, y_{0,3} - y_{1,3}, y_{2,3} - y_{3,3}]$　　（２４）

となる。以上の式（２１）から式（２４）には１６回の差分演算が含まれる。しかしながら、これらには重複する演算が含まれているので、$y_{-2,1} - y_{-1,1}$、$y_{0,1} - y_{-1,1}$、$y_{-2,1} - y_{-3,1}$、$y_{-2,1} - y_{-1,1}$、$y_{0,1} - y_{-1,1}$、$y_{-2,1} - y_{-3,1}$、$y_{-2,3} - y_{-1,3}$、$y_{0,3} - y_{-1,3}$、$y_{-2,3} - y_{-3,3}$で表される９回の差分演算のみを行い、副次参照領域を計算する。この副次参照領域の一部分を取り出すことで、式（２１）から式（２４）の値を生成することができる。

【００８５】このようにして、計算された前処理結果をもとに、$(Y_{u,v-1} + Y_{u+1,v+1})(X_{0,0} - X_{0,-1})$を、（ｕ，ｖ）が、（－２，－２）、（０，－＊２）、（－２，０）、（０，０）について計算する。

【００８６】以上のような計算を式（１６）中の９個の副次内積演算について行い、最終結果$C_{u,v}$を後処理で式（１６）に従って計算する。

【００８７】第５の発明の第２の実施例を示す。上記第１の実施例は式（１６）に基づいて、式（２）の第２項の計算を行った。式（１６）は、式（１３）を式（１４）を用いて変形したものである。本実施例では、式（１４）の代わりに、

【００８８】

【数２１】

$$\begin{bmatrix} A & B \\ B & C \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} (A-B)X + B(X+Y) \\ (C-B)Y + B(X+Y) \end{bmatrix} \quad (25)$$

【００８９】を用いて式（１３）を変形した以下の式、

【数２２】

【００９０】

25 26

$$C_{u,v}=\begin{bmatrix}(Y_{u,v}-Y_{u+1,v}-Y_{u,v+1}+Y_{u+1,v+1})X_{0,0}\\(-Y_{u+1,v}+Y_{u+2,v}+Y_{u+1,v+1}-Y_{u+2,v+1})X_{1,0}\\(-Y_{u,v+1}+Y_{u+1,v+1}+Y_{u,v+2}-Y_{u+1,v+2})X_{0,1}\\(Y_{u+1,v+1}-Y_{u+2,v+1}-Y_{u+1,v+2}+Y_{u+2,v+2})X_{1,1}\end{bmatrix}$$

$$+\begin{bmatrix}1&0\\0&1\\1&0\\0&1\end{bmatrix}\begin{bmatrix}(Y_{u,v+1}-Y_{u+1,v+1})(X_{0,0}+X_{0,1})\\(-Y_{u+1,v+1}+Y_{u+2,v+1})(X_{1,0}+X_{1,1})\end{bmatrix}$$

$$+\begin{bmatrix}1&0\\1&0\\0&1\\0&1\end{bmatrix}\begin{bmatrix}(Y_{u+1,v}-Y_{u+1,v+1})(X_{0,0}+X_{1,0})\\(-Y_{u+1,v+1}+Y_{u+1,v+2})(X_{0,1}+X_{1,1})\end{bmatrix}$$

$$+\begin{bmatrix}1\\1\\1\\1\end{bmatrix}Y_{u+1,v+1}(X_{0,0}+X_{1,0}+X_{0,1}+X_{1,1}) \qquad (26)$$

【００９１】に基づき式（２）の第２項の計算を行う。上記の計算は第１の実施例と同様に行うことができ、演算量も第１の実施例と等しい。

【００９２】第６の発明の実施例として、第５の発明の第１の実施例で説明した副次内積演算、（$Y_{u,v+1}$　＋$Y_{u+1,v+1}$　）（$X_{0,0}$　－$X_{0,1}$　）を更に分割する例を示す。この式を$c'_{u,v}$　と置く。この$c'_{u,v}$　を（ｕ，ｖ）が（－２，－２）、（０，－２）、（－２，０）、（０，０）について計算する必要があった。

【００９３】

【数２３】

$c'_{u,v}$

$$= (Y_{u,v+1} + Y_{u+1,v+1})(X_{0,0} - X_{0,1})$$

$$= \sum_{i=0}^{1}\sum_{j=0}^{1}(x_{2i,2j} - x_{2i,2j+1})(y_{2i+u,2j+v+1} + y_{2i+u+1,2j+v+1}) \quad (27)$$

【００９４】ここで、$x'_{i,j}$、$y'_{i,j}$、u'、v'を以下のように定義する。

$$x'_{i,j} = (x_{2i,2j} - x_{2i,2j+1}) \quad (28)$$

$$y'_{i,j} = (y_{2i,2j+1} - y_{2i+1,2j+1}) \quad (29)$$

$$u' = u/2 \quad (30)$$

$$v' = v/2 \quad (31)$$

これらを用いると、式（２７）は次のように表せる。

【００９５】＊　＊【００９６】※　※【数２４】

$$c'_{2u',2v'} = \sum_{i=0}^{1}\sum_{j=0}^{1} x'_{i,j}y'_{i+u',j+v'} \quad (32)$$

【００９７】この式をu′，v′＝－１～０について計算することになる。この式は式（８）と同様の形式なので、第５の発明と同様の方法で式（１６）のように分割し、演算量を削減することができる。

【００９８】図６は第７の発明による動き検出方式の処理過程を示す図である。第７の発明による動き検出方式は、ある基準参照ブロックの周辺の参照ブロックを探索する場合に適用できるものである。ステップ７０で、周辺の参照ブロックについて式（１７）を計算する際に必要となる第３項を全て計算し記憶する。この計算方法については後で詳しく説明する。ステップ７１で、式（１７）の第１項を積和演算器にて計算し記憶する。この式（１７）の第１項は現ブロックと基準参照ブロックとの内積に－２をかけたものであり、式（１７）の第２項が現ブロックと参照ブロックとの内積を２倍したものであることから、以前の探索において、この項が計算済みの場合はこの計算を簡略化することができる。ステップ７２で最初に探索する動きベクトル、つまり参照ブロックを設定する。またステップ７４で用いる最小値をクリアしておく。次に、ステップ７３～ステップ７７で各参照ブロックを探索する。ステップ７３では、現ブロックと参照ブロックとの内積を２倍したものである式（１７）の第２項を積和演算器にて計算し、既に計算済みの第１項及び、対応する第３項と算術論理演算器にて加算する。ステップ７４で、ステップ７３の結果が探索してきた参照ブロック中で最小値であるかを判定する。ただし、ステップ７３の結果が正の値であったときには、この参照ブロックよりも基準参照ブロックの方が適していることを意味するので最小値とは判断しない。最小値であった場合にはステップ７５で最小値を更新し、このときの動きベクトルを記憶しておく。ステップ７６では、探索する参照ブロックについて全て探索が終了したかを判定し、未了であればステップ７７で次の動きベクトルを設定してステップ７３に戻る。終了していれば、ステップ７８で、最小値に対応する動きベクトルを出力し、対応する参照ブロックと現ブロックとの差分の自乗和を式（１７）から計算して出力する。また、計算済みの現ブロックと対応する参照ブロックの内積値を２倍したものも出力する。これにより、この参照ブロックを基準ブロックとして、さらに周辺を探索する際に、式（１７）の第１項の計算を簡略化することができる。ステップ７０の計算についてさらに詳しく説明する。例として、ブロックサイズが４×４で動きベクトルを水平方向、垂直方向共に±１の範囲で探索する場合を考える。この場合、図１２のように、自乗値の計算が必要な画素が存在する。これを図７のようにＰ１～Ｐ１６、Ｑ１～Ｑ８の領域に分割して、各領域内の画素値の自乗和をまず積和演算器にて計算する。この例では領域Ｐ１内には画素は１個しか存在しないので、この部分の自乗和は一つの画素の自乗和になる。例えば、動きベクトルを±４の範囲で探索する場合には、領域Ｐ１は４×４のブロックになる。これら自乗和を組み合わせることにより、式（１

７）の第３項を計算する。例えば、図１２のように基準参照ブロックからみて動きベクトルが（１，－１）の場＊合の式（１７）の第３項は、

$$(Q1+Q8+Q7+Q6+Q5)-(P3+P4+P5+P6+P7) \quad (33)$$

のように算術論理演算器にて計算する。例えば、ブロックのサイズを１６×１６、フレームのサイズが１７６×１４４であり、動き検出の範囲を－７～７としこの範囲内の参照ブロックをロガリズミックサーチ法を用いて、まず動きベクトルが±４、次に±２、次に±１のように探索する場合を考える。積和演算、算術演算共に１サイクルで実行するプロセッサの場合、式（１）を計算して評価する従来法では、１フレーム分の動きを検出するのに、1.27×10^{6} サイクル必要である。本発明を用いた手法で同様の動き検出を実行する場合、１フレーム分の動きを検出するのに、0.753×10^{6} サイクルで実現可能である。これは従来法の５９．４％の演算量に相当する。

【００９９】

【発明の効果】以上に説明したように本発明によれば、動画像の動きを検出する際に、式（１）を展開して計算するようにしたので、各項の演算量を大幅に削減したり、各項のデータを他に有効に用いたりすることができるという効果が得られる。

【図面の簡単な説明】

【図１】第１、第３の発明の実施例を示すブロック図。

【図２】第２、第３の発明の実施例を示すブロック図。

【図３】第１の発明の実施例を示すフローチャート。　※

※【図４】第２の発明の実施例を示すフローチャート。

【図５】第３の発明の実施例を示すフローチャート。

【図６】第７の発明の実施例を示すフローチャート。

【図７】第７の発明において参照ブロックの画素値の自乗和の計算方法を説明する図。

【図８】第１の発明の作用を説明する図。

【図９】第３の発明の作用を説明する図。

【図１０】第５の発明の作用を説明するブロック図。

【図１１】第７の発明の作用を説明する図。

【図１２】第７の発明の作用を説明する図。

【図１３】動き検出方式を説明する図。

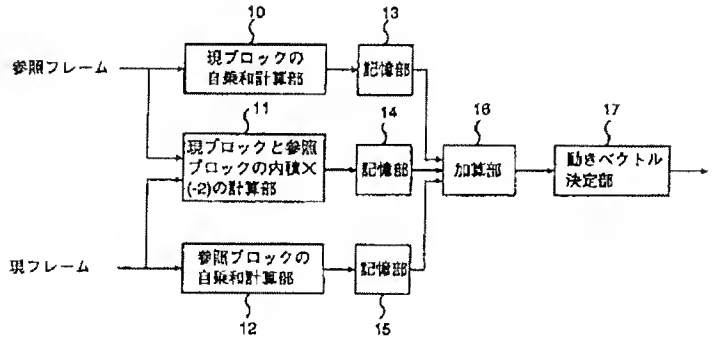
【図１４】積和演算器を有するプロセッサの例。

【図１５】従来法のフローチャート。

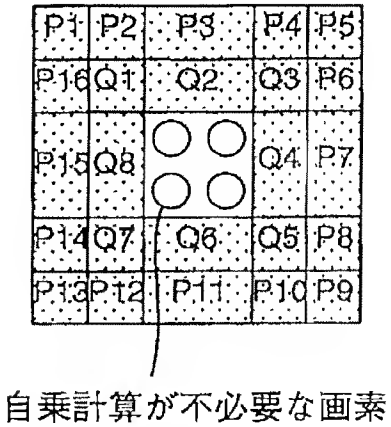
【符号の説明】

１０　現ブロックの自乗和計算部
１１　現ブロックと参照ブロックの内積計算部
１２　参照ブロックの自乗和計算部
１３　現ブロックの自乗和の記憶部
１４　現ブロックと参照ブロックの内積の記憶部
１５　参照ブロックの自乗和の記憶部
１６　加算部
１７　動きベクトル決定部

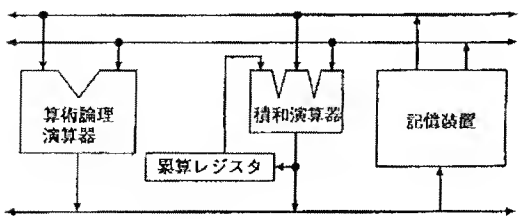
【図１】



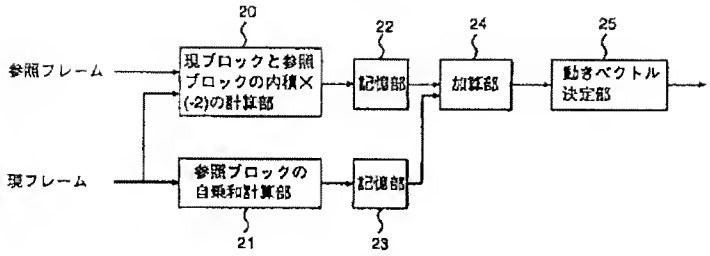
【図７】



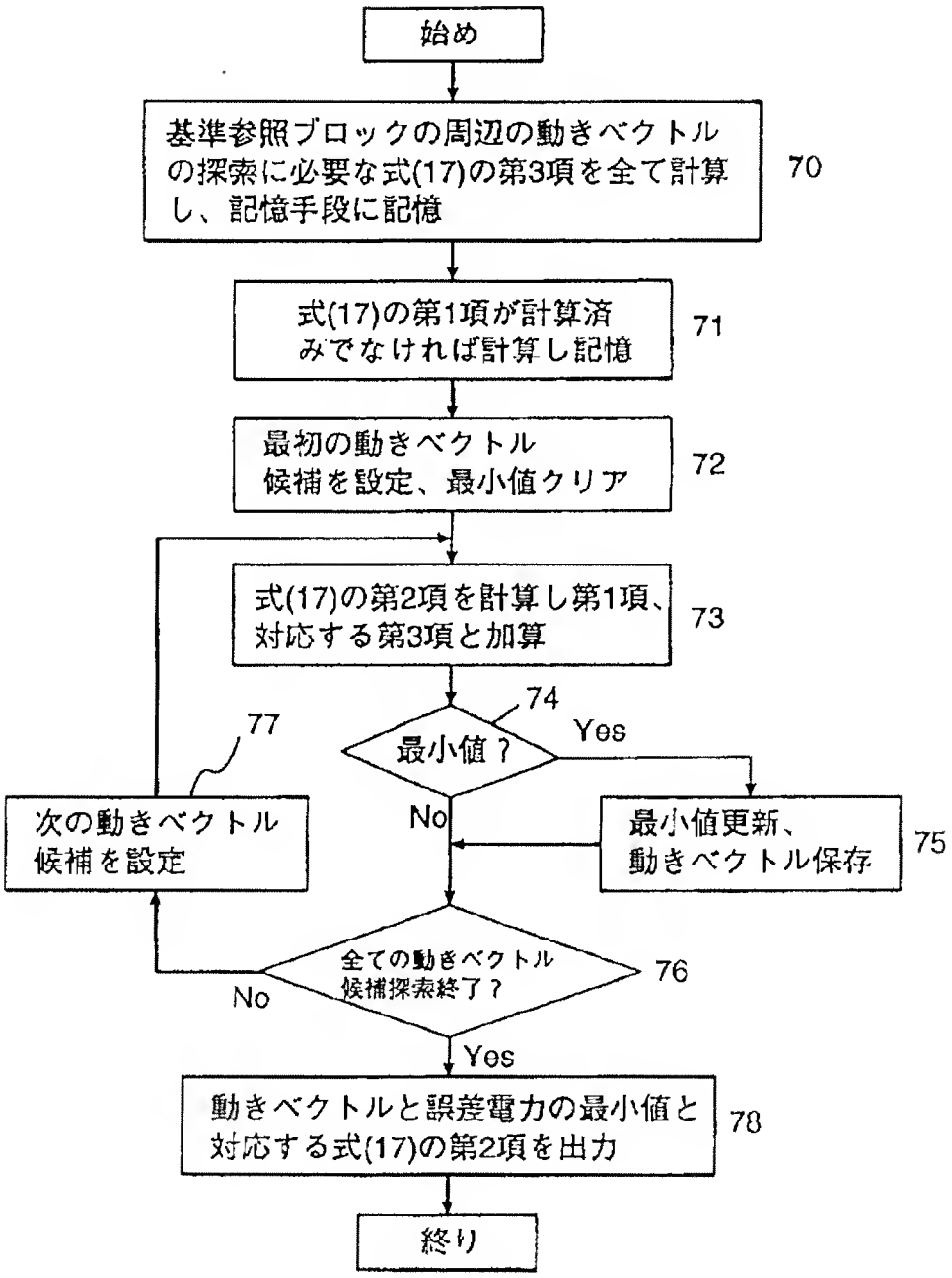
【図１４】



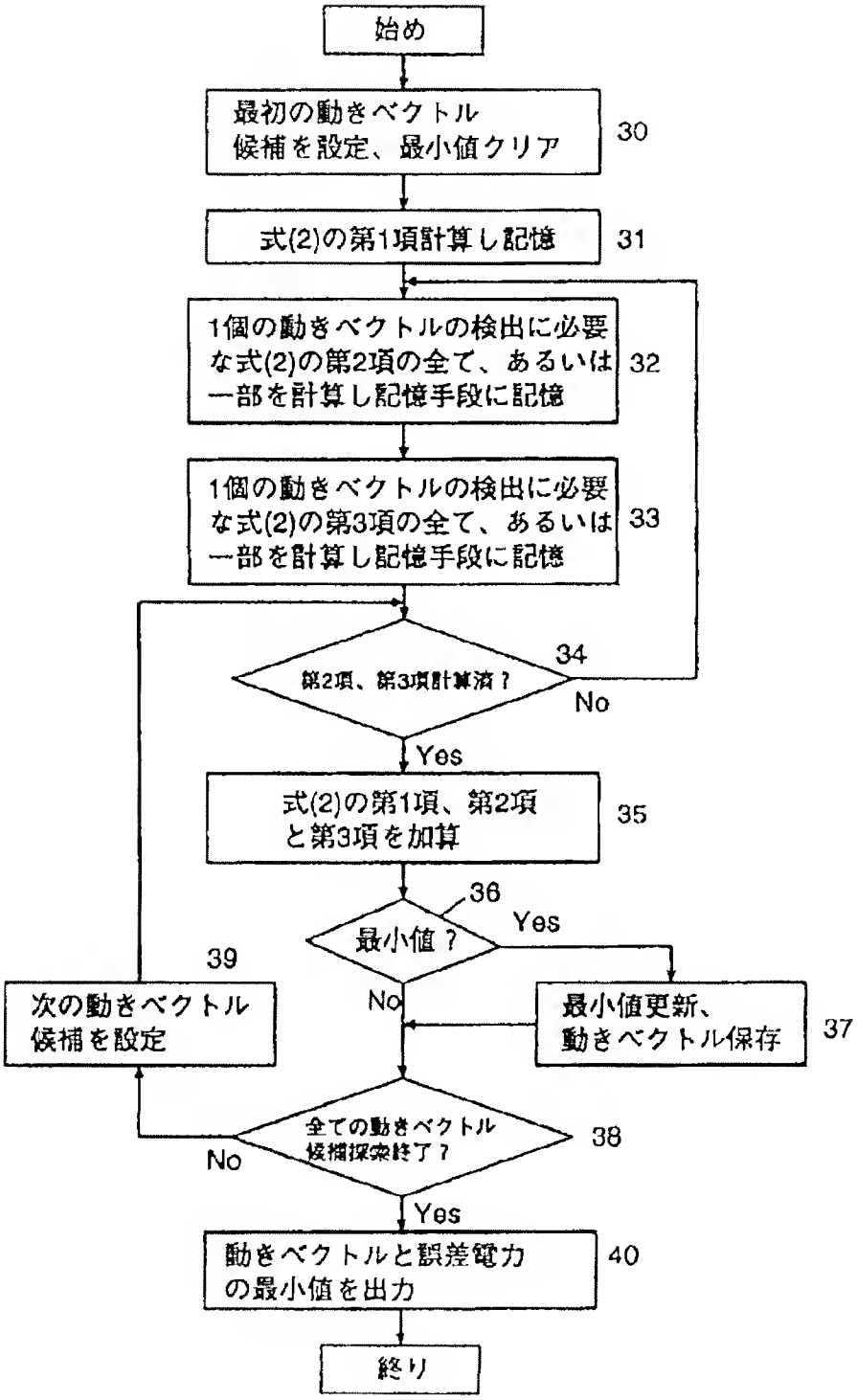
【図２】


20
22
24
25
参照フレーム
現ブロックと参照ブロックの内積×(-2)の計算部
記憶部
加算部
動きベクトル決定部
現フレーム
参照ブロックの自乗和計算部
記憶部
21
23

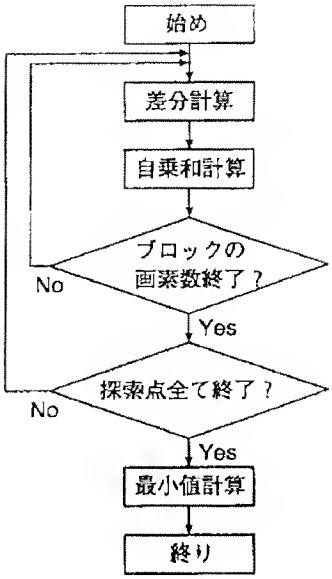
【図６】


始め
基準参照ブロックの周辺の動きベクトルの探索に必要な式(17)の第3項を全て計算し、記憶手段に記憶
70
式(17)の第1項が計算済みでなければ計算し記憶
71
最初の動きベクトル候補を設定、最小値クリア
72
式(17)の第2項を計算し第1項、対応する第3項と加算
73
74
最小値？
Yes
No
77
次の動きベクトル候補を設定
最小値更新、動きベクトル保存
75
全ての動きベクトル候補探索終了？
76
No
Yes
動きベクトルと誤差電力の最小値と対応する式(17)の第2項を出力
78
終り

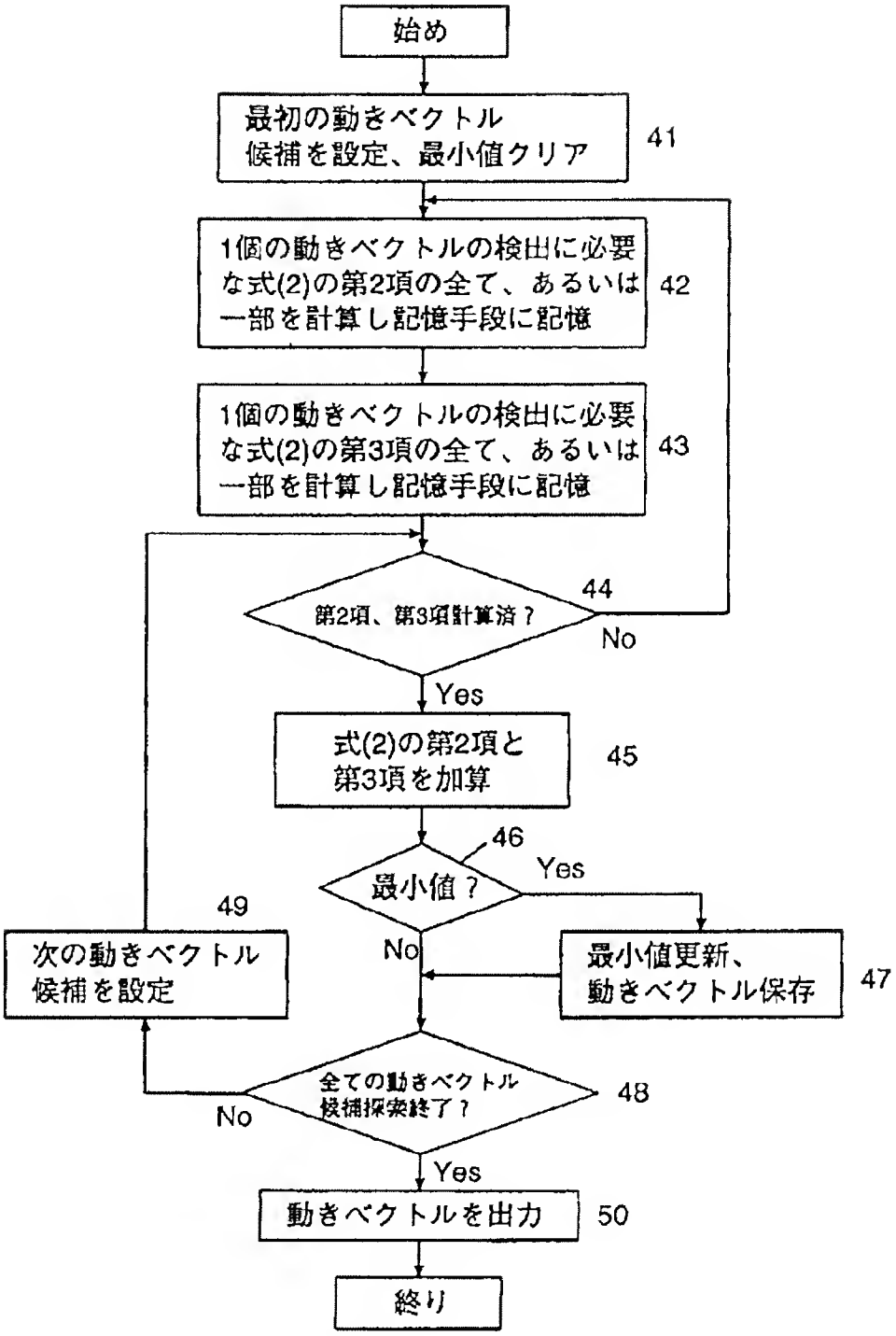
【図３】


始め
最初の動きベクトル候補を設定、最小値クリア
30
式(2)の第1項計算し記憶
31
1個の動きベクトルの検出に必要な式(2)の第2項の全て、あるいは一部を計算し記憶手段に記憶
32
1個の動きベクトルの検出に必要な式(2)の第3項の全て、あるいは一部を計算し記憶手段に記憶
33
第2項、第3項計算済？
34
No
Yes
式(2)の第1項、第2項と第3項を加算
35
36
最小値？
Yes
No
39
次の動きベクトル候補を設定
最小値更新、動きベクトル保存
37
全ての動きベクトル候補探索終了？
38
No
Yes
動きベクトルと誤差電力の最小値を出力
40
終り

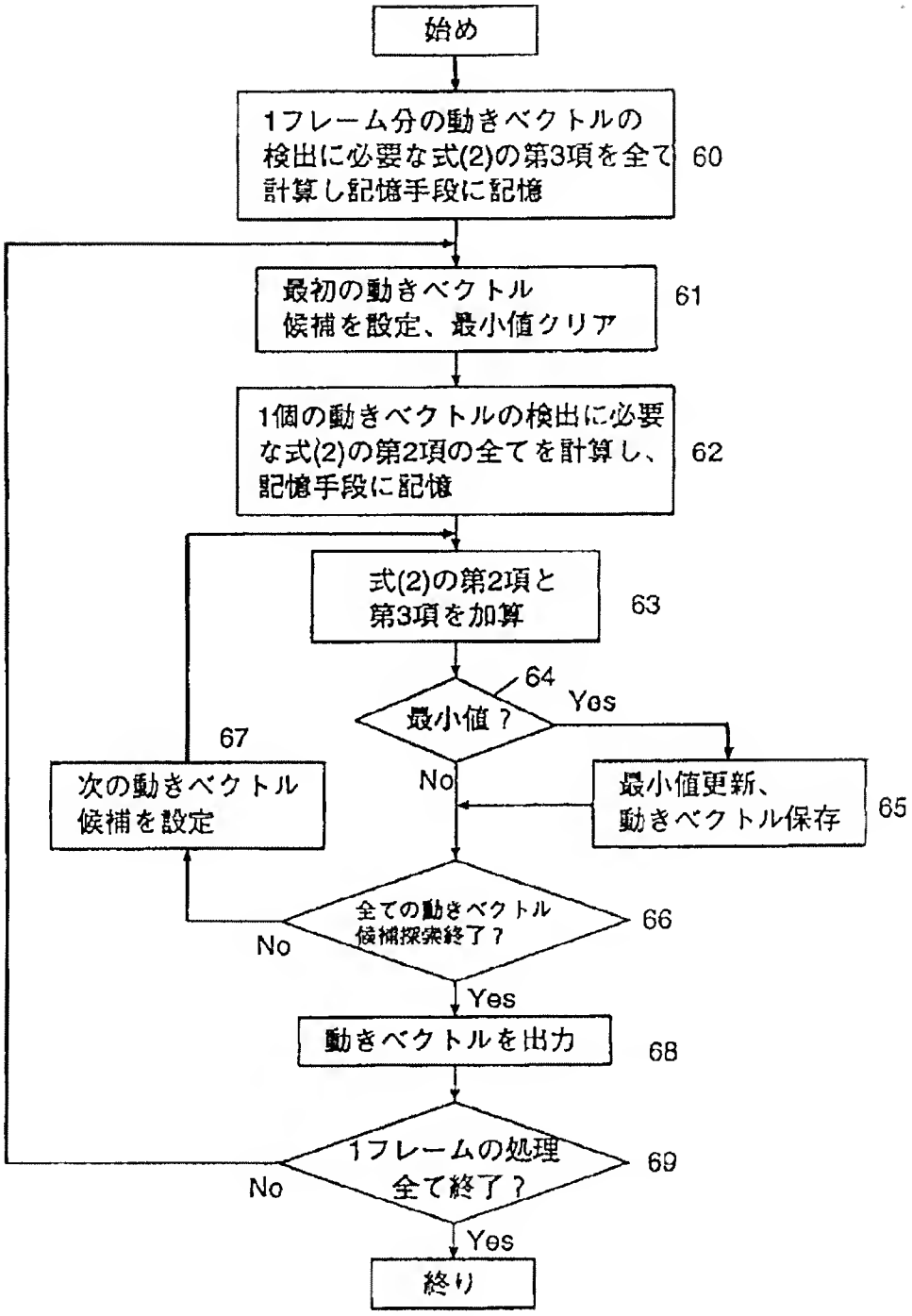
【図１５】


始め
差分計算
自乗和計算
ブロックの画素数終了？
No
Yes
探索点全て終了？
No
Yes
最小値計算
終り

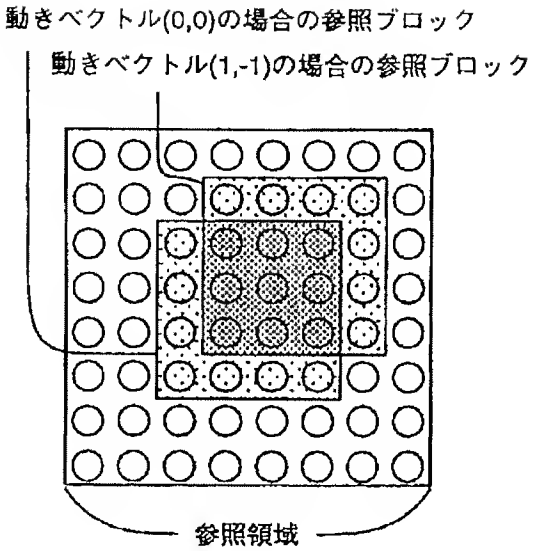
【図４】


始め
最初の動きベクトル候補を設定、最小値クリア
41
1個の動きベクトルの検出に必要な式(2)の第2項の全て、あるいは一部を計算し記憶手段に記憶
42
1個の動きベクトルの検出に必要な式(2)の第3項の全て、あるいは一部を計算し記憶手段に記憶
43
第2項、第3項計算済？
44
No
Yes
式(2)の第2項と第3項を加算
45
最小値？
46
Yes
No
最小値更新、動きベクトル保存
47
次の動きベクトル候補を設定
49
全ての動きベクトル候補探索終了？
48
No
Yes
動きベクトルを出力
50
終り

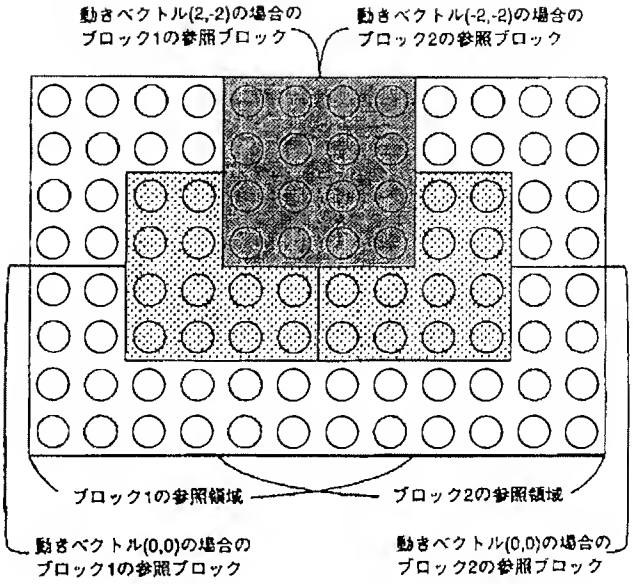
【図５】


始め
1フレーム分の動きベクトルの検出に必要な式(2)の第3項を全て計算し記憶手段に記憶
60
最初の動きベクトル候補を設定、最小値クリア
61
1個の動きベクトルの検出に必要な式(2)の第2項の全てを計算し、記憶手段に記憶
62
式(2)の第2項と第3項を加算
63
64
最小値？
Yes
No
最小値更新、動きベクトル保存
65
67
次の動きベクトル候補を設定
全ての動きベクトル候補探索終了？
66
No
Yes
動きベクトルを出力
68
1フレームの処理全て終了？
69
No
Yes
終り

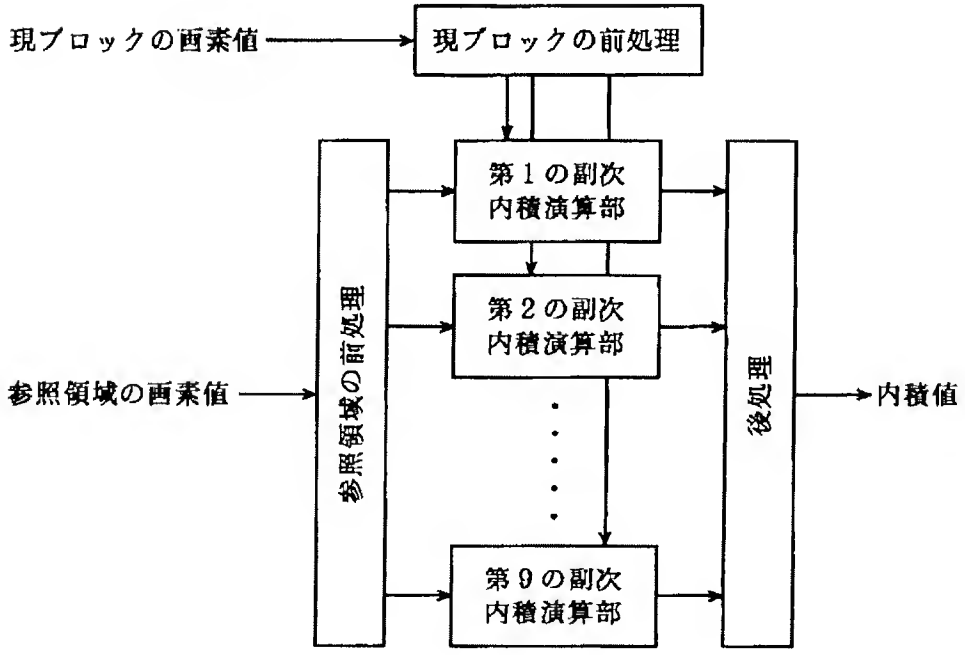
【図８】


動きベクトル(0,0)の場合の参照ブロック
動きベクトル(1,-1)の場合の参照ブロック
参照領域

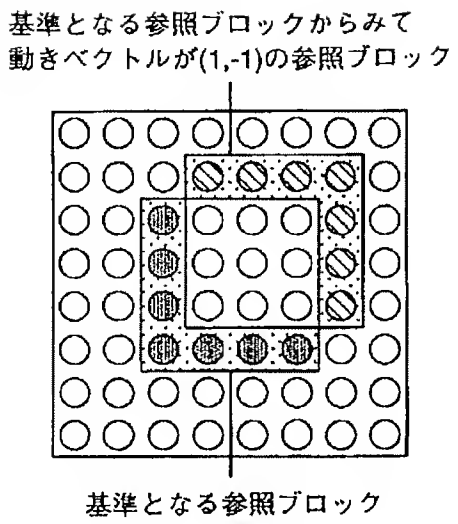
【図９】


動きベクトル(2,-2)の場合の
ブロック1の参照ブロック
動きベクトル(-2,-2)の場合の
ブロック2の参照ブロック
ブロック1の参照領域
ブロック2の参照領域
動きベクトル(0,0)の場合の
ブロック1の参照ブロック
動きベクトル(0,0)の場合の
ブロック2の参照ブロック

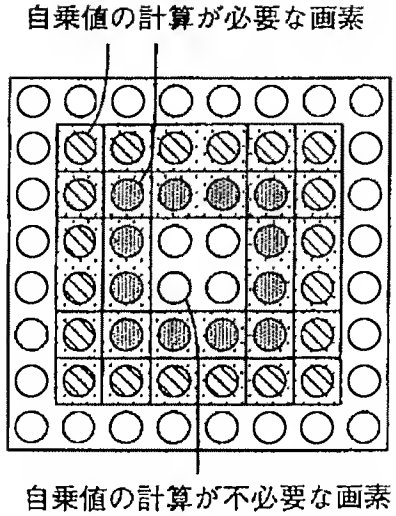
【図１０】


現ブロックの画素値
現ブロックの前処理
第１の副次
内積演算部
第２の副次
内積演算部
第９の副次
内積演算部
参照領域の画素値
参照領域の前処理
後処理
内積値

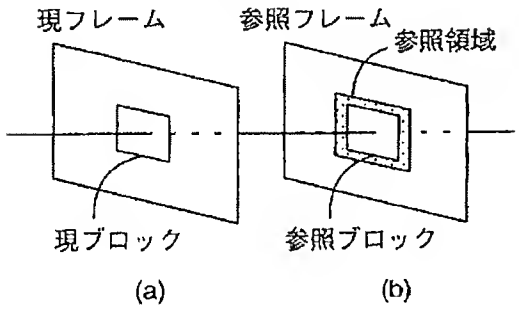
【図１１】


基準となる参照ブロックからみて
動きベクトルが(1,-1)の参照ブロック
基準となる参照ブロック

【図１２】


自乗値の計算が必要な画素
自乗値の計算が不必要な画素

【図１３】


現フレーム
参照フレーム
参照領域
現ブロック
参照ブロック
(a)
(b)